Decreased Idle Time And Constant Bandwidth Data-On-Demand Broadcast Delivery Matrices

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application is a continuation-in-part application of Khoi Hoang's parent application entitled SYSTEMS AND METHODS FOR PROVIDING VIDEO-ON-DEMAND SERVICES FOR BROADCASTING SYSTEMS filed May 31, 2000, bearing application no. 09/584,832, which is incorporated herein by reference. The present application further claims priority to Khoi Nhu Hoang's co-pending patent application entitled UNIVERSAL DIGITAL BROADCAST SYSTEM AND METHODS filed on April 24, 2001, bearing application no. 09/841,792, which is also incorporated herein by reference.

BACKGROUND OF THE INVENTION

Video-on-demand (VOD) systems are one type of data-on-demand (DOD) system. In VOD systems, video data files are provided by a server or a network of servers to one or more clients on a demand basis. These systems will be well understood by those of skill in the art.

In a conventional VOD architecture, a server or a network of servers communicates with clients in a standard hierarchical client-server model. For example, a client sends a request to a server for a data file (e.g., a video data file). In response to the client request, the server sends the requested file to the client. In the standard client-server model, a client's request for a data file can be fulfilled by one or more servers. The client may have the capability to store any received data file locally in non-volatile memory for later use. The standard client-server model requires a two-way

communications infrastructure. Currently, two-way communications requires building new infrastructure because existing cables can only provide one-way communications. Examples of two-way communications infrastructure are hybrid fiber optics coaxial cables (HFC) or all fiber infrastructure. Replacing existing cables is very costly and the resulting services may not be affordable to most users.

In addition, the standard client-server model has many limitations when a service provider (e.g., a cable company) attempts to provide VOD services to a large number of clients. One limitation of the standard client-server model is that the service provider has to implement a mechanism to continuously listen and fulfill every request from each client within the network; thus, the number of clients who can receive service is dependent on the capacity of such a mechanism. One mechanism uses massively-parallel computers having large and fast disk arrays as local servers. However, even the fastest existing local server can only deliver video data streams to about 1000 to 2000 clients at one time. Thus, in order to service more clients, the number of local servers must increase. Increasing local servers requires more upper level servers to maintain control of the local servers.

Another limitation of the standard client-server model is that each client requires its own bandwidth. Thus, the total required bandwidth is directly proportional to the number of subscribing clients. Cache memory within local servers has been used to improve bandwidth limitation but using cache memory does not solve the problem because cache memory is also limited.

Presently, in order to make video-on-demand services more affordable for clients, existing service providers are increasing the ratio of clients per local server above the local server's capabilities. Typically, a local server,

which is capable of providing service to 1000 clients, is actually committed to service 10,000 clients. This technique may work if most of the subscribing clients do not order videos at the same time. However, this technique is set up for failure because most clients are likely to want to view videos at the same time (i.e., evenings and weekends), thus, causing the local server to become overloaded.

Thus, it is desirable to provide a system that is capable of providing on-demand services to a large number of clients over virtually any transmission medium without replacing existing infrastructure.

SUMMARY OF THE INVENTION

In an exemplary embodiment, at a server side, a method for sending data to a client to provide data-on-demand services comprising the steps of: receiving a data file, specifying a time interval, parsing the data file into a plurality of data blocks based on the time interval such that each data block is displayable during the time interval, determining a required number of time slots to send the data file, allocating to each time slot at least a first of the plurality of data blocks and optionally one or more additional data blocks, such that the plurality of data blocks is available in sequential order to a client accessing the data file during any time slot, and sending the plurality of data blocks based on the allocating step. In one embodiment, the parsing step includes the steps of: determining an estimated data block size, determining a cluster size of a memory in a channel server, and parsing the data file based on the estimated data block size and the cluster size. In another embodiment, the determining step includes the step of assessing resource allocation and bandwidth availability.

In an exemplary embodiment, at a client side, a method for processing data received from a server to provide data-on-demand services comprises the steps of: (a) receiving a selection of a data file during a first time slot; (b) receiving at least one data block of the data file during a second time slot; (c) during a next time slot: receiving any data block not already received, sequentially displaying a data block of the data file, and repeating step (c) until all data blocks of the data file has been received and displayed. In one embodiment, the method for processing data received from a server is performed by a set-top box at the client side.

In an exemplary embodiment, a data file is divided into a number of data blocks and a scheduling matrix is generated based on the number of data blocks. At the server side, the scheduling matrix provides a send order for sending the data blocks, such that a client can access the data blocks in sequential order at a random time. In an exemplary embodiment, a method for generating a scheduling matrix for a data file comprises the steps of: (a) receiving a number of data blocks [x] for a data file; (b) setting a first variable [j] to zero; (c) setting a second variable [i] to zero; (d) clearing all entries in a reference array; (e) writing at least one data block stored in matrix positions of a column [(i+j) modulo x] in a matrix to a reference array, if the reference array does not already contain the data block; (f) writing a data block [i] into the reference array and a matrix position [(i+j) modulo x,j] of the matrix, if the reference array does not contain the data block [i]; (g) incrementing the second variable [i] by one and repeating step (e) until the second variable [i] is equal to the number of data blocks [x]; and (h) incrementing the first variable [j] by one and repeating the step (c) until the first variable [j] is equal to the number of data blocks [x]. In one embodiment, a scheduling matrix is generated for each data file in a set of data files and a convolution method is applied to generate a delivery matrix based on the scheduling matrices for sending the set of data files.

A data-on-demand system comprises a first set of channel servers, a central controlling server for controlling the first set of channel servers, a first set of up-converters coupled to the first set of channel servers, a combiner/amplifier coupled to the first set of up-converters, and a combiner/amplifier adapted to transmit data via a transmission medium. In an exemplary embodiment, the data-on-demand system further comprises a channel monitoring module for monitoring the system, a switch matrix, a

second set of channel servers, and a second set of up-converters. The channel monitoring module is configured to report to the central controlling server when system failure occurs. The central controlling server, in response to a report from the channel monitoring module, instructs the switch matrix to replace a defective channel server in the first set of channel servers with a channel server in the second set of channel servers and a defective up-converter in the first set of up-converters with an up-converter in the second set of up-converters.

A method for providing data-on-demand services comprises the steps of calculating a delivery matrix of a data file, sending the data file in accordance with the delivery matrix, such that a large number of clients is capable of viewing the data file on demand. In one embodiment, the data file includes a video file.

In yet another embodiment of the present invent, the idle time created in a delivery matrix can be decreased by moving up the next data blocks in the matrix until all time slots are full and the entire bandwidth is used. In this manner, the delivery matrix can be thought of more as a stream of data maintaining the order of the original matrix. At any time a user may join the stream and being using data-on-demand services as soon as a starting block is received.

Since the matrix can be thought of as a stream, a user can enter the stream at any given time, and wait only as long as it takes to receive a starting block to being using the data-on-demand services, which would be no longer than the predetermined time slots of the original delivery matrix.

Another embodiment of the present invention teaches a universal STB capable of receiving and handling a plurality of digital services such as VOD and digital broadcast. This embodiment teaches a universal STB having a

highly flexible architecture capable of sophisticated processing of received data. This architecture includes a databus, a first communication device suitable for coupling to a digital broadcast communications medium, a memory typically including persistent and transient memory bi-directionally coupled to the databus, a digital data decoder bi-directionally coupled to the databus, and a central processing unit (CPU) bi-directionally coupled to the databus. The CPU of this embodiment of the present invention implements a STB control process for controlling the memory, the digital decoder, and the demodulator. The STB control process is operable to process digital data such as that received at the first communications device. The STB control process should be capable of receiving data blocks derived from a decreased idle time scheduling matrix as well as parallel streaming of such data blocks.

BRIEF DESCRIPTION OF THE DRAWINGS

FIGURE 1A illustrates an exemplary DOD system in accordance with an embodiment of the invention.

FIGURE 1B illustrates an exemplary DOD system in accordance with another embodiment of the invention.

FIGURE 2 illustrates an exemplary channel server in accordance with an embodiment of the invention.

FIGURE 3 illustrates an exemplary set-top box in accordance with an embodiment of the invention.

FIGURE 4 illustrates an exemplary process for generating a scheduling matrix in accordance with an embodiment of the invention.

FIGURE 5 graphically illustrates an example of a scheduling matrix of a six data block file.

FIGURE 6 graphically illustrates how the data blocks of the scheduling matrix in Figure 6 are moved up until all idle time slots are filled.

FIGURE 7 graphically illustrates a new decreased idle time scheduling matrix.

FIGURE 8 depicts the addition of the decreased idle time embodiment.

FIGURE 9 shows in flow chart form how the decreased idle time embodiment is accomplished.

FIGURE 10 graphically depicts multiple stream of repeating data being created from an original scheduling matrix.

DETAILED DESCRIPTION OF THE INVENTION

Figure 1A illustrates an exemplary DOD system 100 in accordance with an embodiment of the invention. In this embodiment, the DOD system 100 provides data files, such as video files, on demand. However, the DOD system 100 is not limited to providing video files on demand but is also capable of providing other data files, for example, game files on demand. The DOD system 100 includes a central controlling server 102, a central storage 103, a plurality of channel servers 104a-104n, a plurality of up-converters 106a-106n, and a combiner/amplifier 108. The central controlling server 102 controls the channel servers 104. The central storage 103 stores data files in digital format. In an exemplary embodiment, data files stored in the central storage 103 are accessible via a standard network interface (e.g., Ethernet connection) by any authorized computer, such as the central controller server 102, connected to the network. Each channel server 104 is assigned to a channel and is coupled to an up-converter 106. The channel servers 104 provide data files that are retrieved from the central storage 103 in accordance with instructions from the central controlling server 102. The output of each channel server 104 is a quadrature amplitude modulation (QAM) modulated intermediate frequency (IF) signal having a suitable frequency for the corresponding up-converter 106. The QAM-modulated IF signals are dependent upon adopted standards. The current adopted standard in the United States is the data-over-cable-systems-interface-specification (DOCSIS) standard, which requires an approximately 43.75MHz IF frequency. The up-converters 106 convert IF signals received from the channel servers 104 to radio frequency signals (RF signals). The RF signals, which include frequency and bandwidth, are dependent on a

desired channel and adopted standards. For example, under the current standard in the United States for a cable television channel 80, the RF signal has a frequency of approximately 559.25MHz and a bandwidth of approximately 6MHz. The outputs of the up-converters 106 are applied to the combiner/amplifier 108. The combiner/amplifier 108 amplifies, conditions, and combines the received RF signals then outputs the signals out to a transmission medium 110.

In an exemplary embodiment, the central controlling server 102 includes a graphics user interface (not shown) to enable a service provider to schedule data delivery by a drag-and-drop operation. Further, the central controlling server 102 authenticates and controls the channel servers 104 to start or stop according to delivery matrices. In an exemplary embodiment, the central controlling server 102 automatically selects a channel and calculates delivery matrices for transmitting data files in the selected channel. The central controlling server 102 provides offline addition, deletion, and update of data file information (e.g., duration, category, rating, and/or brief description). Further, the central controlling server 102 controls the central storage 103 by updating data files and databases stored therein.

In an exemplary embodiment, an existing cable television system 120 may continue to feed signals into the combiner/amplifier 108 to provide non-DOD services to clients. Thus, the DOD system 100 in accordance with the invention does not disrupt present cable television services.

Figure 1B illustrates another exemplary embodiment of the DOD system 100 in accordance with the invention. In addition to the elements illustrated in Figure 1A, the DOD system 100 includes a switch matrix 112, a channel monitoring module 114, a set of back-up channel servers 116a-116b, and a set of back-up up-converters 118a-118b. In one embodiment,

the switch matrix 112 is physically located between the up-converters 106 and the combiner/amplifier 108. The switch matrix 112 is controlled by the central controlling server 102. The channel monitoring module 114 comprises a plurality of configured set-top boxes, which simulate potential clients, for monitoring the health of the DOD system 100. Monitoring results are communicated by the channel monitoring module 114 to the central controlling server 102. In case of a channel failure (i.e., a channel server failure, an up-converter failure, or a communication link failure), the central controlling server 102 through the switch matrix 112 disengages the malfunctioning component and engages a healthy backup component 116 and/or 118 to resume service.

In an exemplary embodiment, data files being broadcasted from the DOD system 100 are contained in motion pictures expert group (MPEG) files. Each MPEG file is dynamically divided into data blocks and sub-blocks mapping to a particular portion of a data file along a time axis. These data blocks and sub-blocks are sent during a pre-determined time in accordance with three-dimensional delivery matrices provided by the central controlling server 102. A feedback channel is not necessary for the DOD system 100 to provide DOD services. However, if a feedback channel is available, the feedback channel can be used for other purposes, such as billing or providing Internet services.

Figure 2 illustrates an exemplary channel server 104 in accordance with an embodiment of the invention. The channel server 104 comprises a server controller 202, a CPU 204, a QAM modulator 206, a local memory 208, and a network interface 210. The server controller 202 controls the overall operation of the channel server 104 by instructing the CPU 204 to divide data files into blocks (further into sub-blocks and data packets), select

data blocks for transmission in accordance with a delivery matrix provided by the central controlling server 102, encode selected data, compress encoded data, then deliver compressed data to the QAM modulator 206. The QAM modulator 206 receives data to be transmitted via a bus (i.e., PCI, CPU local bus) or Ethernet connections. In an exemplary embodiment, the QAM modulator 206 may include a downstream QAM modulator, an upstream quadrature amplitude modulation/quadrature phase shift keying (QAM/QPSK) burst demodulator with forward error correction decoder, and/or an upstream tuner. The output of the QAM modulator 206 is an IF signals that can be applied directly to an up-converter 106.

The network interface 210 connects the channel server 104 to other channel servers 104 and to the central controlling server 102 to execute the scheduling and controlling instructions from the central controlling server 102, reporting status back to the central controlling server 102, and receiving data files from the central storage 103. Any data file retrieved from the central storage 103 can be stored in the local memory 208 of the channel server 104 before the data file is processed in accordance with instructions from the server controller 202. In an exemplary embodiment, the channel server 104 may send one or more DOD data streams depending on the bandwidth of a cable channel (e.g., 6, 6.5, or 8MHz), QAM modulation (e.g., QAM 64 or QAM 256, and a compression standard/bit rate of the DOD data stream (i.e., MPEG-1 or MPEG-2).

FIG. 3 illustrates a universal set-top box (STB) **300** in accordance with one embodiment of the invention. The STB **300** comprises a QAM demodulator **302**, a CPU **304**, a local memory **308**, a buffer memory **310**, a decoder **312** having video and audio decoding capabilities, a graphics overlay module **314**, a user interface **318**, a communications link **320**, and a

fast data bus **322** coupling these devices as illustrated. The CPU **302** controls overall operation of the universal STB **300** in order to select data in response to a client's request, decode selected data, decompress decoded data, re-assemble decoded data, store decoded data in the local memory **308** or the buffer memory **310**, and deliver stored data to the decoder **312**. In an exemplary embodiment, the local memory **308** comprises non-volatile memory (e.g., a hard drive) and the buffer memory **310** comprises volatile memory.

In one embodiment, the QAM demodulator **302** comprises transmitter and receiver modules and one or more of the following: privacy encryption/decryption module, forward error correction decoder/encoder, tuner control, downstream and upstream processors, CPU and memory interface circuits. The QAM demodulator **302** receives modulated IF signals, samples and demodulates the signals to restore data.

In an exemplary embodiment, when access is granted, the decoder **312** decodes at least one data block to transform the data block into images displayable on an output screen. The decoder **312** supports commands from a subscribing client, such as play, stop, pause, step, rewind, forward, etc. The decoder **312** provides decoded data to an output device **324** for use by the client. The output device **324** may be any suitable device such as a television, computer, any appropriate display monitor, a VCR, or the like.

The graphics overlay module **314** enhances displayed graphics quality by, for example, providing alpha blending or picture-in-picture capabilities. In an exemplary embodiment, the graphics overlay module **314** can be used for graphics acceleration during game playing mode, for example, when the service provider provides games-on-demand services using the system in accordance with the invention.

The user interface **318** enables user control of the STB **300**, and may be any suitable device such as a remote control device, a keyboard, a smartcard, etc. The communications link **320** provides an additional communications connection. This may be coupled to another computer, or may be used to implement bi-directional communication. The data bus **322** is preferably a commercially available "fast" data bus suitable for performing data communications in a real time manner as required by the present invention. Suitable examples are USB, firewire, etc.

In an exemplary embodiment, although data files are broadcast to all cable television subscribers, only the DOD subscriber who has a compatible STB **300** will be able to decode and enjoy data-on-demand services. In one exemplary embodiment, permission to obtain data files on demand can be obtained via a smart card system in the user interface **318**. A smart card may be rechargeable at a local store or vending machine set up by a service provider. In another exemplary embodiment, a flat fee system provides a subscriber unlimited access to all available data files.

In preferred embodiments, data-on-demand interactive features permits a client to select at any time an available data file. The amount of time between when a client presses a select button and the time the selected data file begins playing is referred to as a response time. As more resources are allocated (e.g., bandwidth, server capability) to provide DOD services, the response time gets shorter. In an exemplary embodiment, a response time can be determined based on an evaluation of resource allocation and desired quality of service. When combined with the embodiment of placing the first data block in a parallel stream, the response time becomes a factor only of the time it takes to receive and process that first data block.

In one embodiment, the number of data blocks (NUM_OF_BLKS) for each data file can be calculated as follows:

Estimated_BLK_Size = (DataFile Size * TS) / DataFile_Length (1)

BLK SIZE = (Estimated BLK Size + CLUSTER SIZE - 1Byte) / CLUSTER _SIZE (2)

BLK_SIZE_BYTES = BLK_SIZE * CLUSTER_SIZE (3)

NUM_OF_BLKS =(DataFile_Size + BLK_SIZE_BYTES - 1Byte)/BLK_SIZE_BYTES (4)

In equations (1) to (4), the Estimated_BLK_Size is an estimated block size (in Bytes); the DataFile_Size is the data file size (in Bytes); TS represents the duration of a time slot (in seconds); DataFile_Length is the duration of the data file (in seconds); BLK SIZE is the number of clusters needed for each data block; CLUSTER _SIZE is the size of a cluster in the local memory 208 for each channel server 104 (e.g., 64KBytes); BLK_SIZE_BYTES is a block size in Bytes. In this embodiment, the number of blocks (NUM_OF_BLKS) is equal to the data file size (in Bytes) plus a data block size in Bytes minus l, Byte and divided by a data block size in Bytes. Equations (1) to (4) illustrate one specific embodiment. A person of skill in the art would recognize that other methods are available to calculate a number of data blocks for a data file. For example, dividing a data file into a number of data blocks is primarily a function of an estimated block size and the cluster size of the local memory 208 of a channel server 104. Thus, the invention should not be limited to the specific embodiment presented above.

Figure 4 illustrates an exemplary process for generating a scheduling matrix for sending a data file in accordance with an embodiment of the invention. In an exemplary embodiment, this invention uses time division multiplexing (TDM) and frequency division multiplexing (FDM) technology to compress and schedule data delivery at the server side. In an exemplary embodiment, a scheduling matrix is generated for each data file. In one embodiment, each data file is divided into a number of data blocks and the scheduling matrix is generated based on the number of data blocks. Typically, a scheduling matrix provides a send order for sending data blocks of a data file from a server to clients, such that the data blocks are accessible in sequential order by any client who wishes to access the data file at a random time.

At step 402, a number of data blocks (x) for a data file is received. A first variable, j, is set to zero (step 404). A reference array is cleared (step 406). The reference array keeps track of data blocks for internal management purposes. Next, j is compared to x (step 408). If j is less than x, a second variable, i, is set to zero (step 412). Next, i is compared to x (step 414). If i is less than x, data blocks stored in the column [(i+j) modulo (x)] of a scheduling matrix are written into the reference array (step 418). If the reference array already has such data block(s), do not write a duplicate copy. Initially, since the scheduling matrix does not yet have entries, this step can be skipped. Next, the reference array is checked if it contains data block i (step 420). Initially, since all entries in the reference array have been cleared at step 406, there would be nothing in the reference array. If the reference array does not contain data block i, data block i is added into the scheduling matrix at matrix position [(i+j) modulo (x), j] and the reference array (step 422). After the data block i is added to the scheduling matrix and the

reference array, i is incremented by 1, such that i = i + 1 (step 424), then the process repeats at step 414 until i = x. If the reference array contains data block i, i is incremented by 1, such that i = i + 1 (step 424), then the process repeats at step 414 until i = x. When i = x, j is incremented by 1, such that j = j + 1 (step 416) and the process repeats at step 406 until j = x. The entire process ends when j = x (step 410).

In an exemplary embodiment, if a data file is divided into six data blocks (x = 6), the scheduling matrix and the reference arrays are as follows:

Scheduling Matrix (SM)

TS0	TS1	TS2	TS3	TS4	TS5
[0, 0] blk0	[1,0] blk1	[2, 0] blk2	[3, 0] blk3	[4, 0] blk4	[5, 0] blk5
[0, 1]	[1, 1] blk0	[2, 1]	[3, 1]	[4, 1]	[5, 1]
[0, 2]	[1, 2]	[2, 2] blk0	[3, 2] blk1	[4, 2]	[5, 2]
[0, 3]	[1, 3]	[2, 3]	[3, 3] blk0	[4, 3]	[5, 3] blk2
[0, 4]	[1, 4] blk3	[2, 4]	[3, 4]	[4, 4] blk0	[5, 5] blk1
[0, 5]	[1, 5]	[2, 5]	[3, 5] blk4	[4, 5]	[5, 5] blk0

Reference Array (RA)

	space0	space1	space2	space3	space4	space5
TS0	blk0	blk1	blk2	blk3	blk4	blk5
TS1	blk1	blk0	blk2	blk3	blk4	blk5
TS2	blk2	blk0	blk3	blk1	blk4	blk5

TS3	blk3	blk1	blk0	blk4	blk5	blk2
TS4	blk4	blk0	blk5	blk2	blk1	blk3
TS5	blk5	blk2	blk1	blk0	blk3	blk4

Appendix A attached to this application describes a step-by-step process of the exemplary process illustrated in Figure 4 to generate the above scheduling matrix and reference arrays. In this exemplary embodiment, based on the scheduling matrix above, the six data blocks of the data file are sent in the following sequence:

TS0 => blk0
TSI => blk0, blkl, blk3
TS2 => blk0, blk2
TS3 => blk0, blkl, blk3, blk4
TS4 => blk0, blk4
TS5 => blk0, blkl, blk2, blk5

In another exemplary embodiment, a look-ahead process can be used to calculate a look-ahead scheduling matrix to send a predetermined number of data blocks of a data file prior to a predicted access time. For example, if a predetermined look-ahead time is the duration of one time slot, for any time slot greater than or equal to time slot number four, data block 4 (blk4) of a data file should be received by a STB 300 at a subscribing client at or before TS3, but blk4 would not be played until TS4. The process steps for generating a look-ahead scheduling matrix is substantially similar to the process steps described above for Figure 4 except that the look-ahead scheduling matrix in this embodiment schedules an earlier sending sequence

based on a look-ahead time. Assuming a data file is divided into six data blocks, an exemplary sending sequence based on a look-ahead scheduling matrix, having a look-ahead time of the duration of two time slots, can be represented as follows:

TS0 => blk0

TS1 => blk0, blk1, blk3, blk4

TS2 => blk0, blk2

TS3 => blk0, blk1, blk3, blk4, blk5

TS4 => blk0, blk5

TS5 => blk0, blk1, blk2

A three-dimensional delivery matrix for sending a set of data files is generated based on the scheduling matrices for each data file of the set of data files. In the three-dimensional delivery matrix, a third dimension containing IDs for each data file in the set of data files is generated. The three-dimensional delivery matrix is calculated to efficiently utilize available bandwidth in each channel to deliver multiple data streams. In an exemplary embodiment, a convolution method, which is well known in the art, is used to generate a three-dimensional delivery matrix to schedule an efficient delivery of a set of data files. For example, a convolution method may include the following policies: (1) the total number of data blocks sent in the duration of any time slot (TS) should be kept at a smallest possible number; and (2) if multiple partial solutions are available with respect to policy (1), the preferred solution is the one which has a smallest sum of data blocks by adding the data blocks to be sent during the duration of any reference time slot, data blocks to be sent during the duration of a previous

time slot (with respect to the reference time slot), and data blocks to be sent during the duration of a next time slot (with respect to the reference time slot). For example, assuming an exemplary system sending two short data files, M and N, where each data file is divided into six data blocks, the sending sequence based on a scheduling matrix is as follows:

TS0 => blk0

TS1 => blk0, blkl, blk3, blk4

TS2 => blk0, blk2

TS3 => blk0, blk1, blk3, blk4

TS4 => blk0, blk4

TS5 => blk0,blkl,blk2, blk5

Applying the exemplary convolution method as described above, possible combinations of delivery matrices are as follows:

Option 1: Send video file N at shift 0 TS	Total Data Blocks
TS0 => M0, N0	2
TS1 => M0,M1,M3,N0,NI,N3	6
TS2 => M0, M2, N0, N2	4
TS3 => M0, Ml, M3, M4, N0, N1, N3, N4	8
TS4 => M0, M4, N0, N4	4
TS5 => M0, MI, M2, M5, N0, NI, N2, N5	8

Option 2: Send video file N at shift 1 TS	Total Data Blocks
TS0 => M0, N0, N1, N3	4
TS1 => M0, MI, M3, N0, N2	5
TS2 => M0, M2, N0, N1, N3, N4	6
TS3 =>M0, Ml, M3, M4, N0, N4	6
TS4 => M0, M4, N0, N1, N2, N5	6
TS5 => M0, Ml, M2, M5, N0	5

Option 3: Send video file N at shift 2 TS	Total Data Blocks
TS0 => M0, N0, N2	3
TS1 => M0, M1, M3, N0, N1, N3, N4	7
TS2 => M0, M2, N0, N4	4
TS3 =>M0, Ml, M3, M4, N0, N1, N2, N5	8
TS4 => M0, M4, N0	3
TS5 => M0, Ml, M2, M5, N0, N1, N3	7

Option 4: Send video file N at shift 3 TS	Total Data Blocks
TS0 => M0, N0, N1, N3, N4	5
TS1 => M0, M1, M3, N0, N4	5
TS2 => M0, M2, N0, N1, N2, N5	6
TS3 =>M0, Ml, M3, M4, N0	5

TS4 => M0, M4, N0, N1, N3

TS5 => M0, Ml, M2, M5, N0, N1, N2

Option 5: Send video file N at shift 4 TS	Total Data Blocks
TS0 =>M0, N0, N4	3
TS1 => M0, Ml, M3, N0, N1, N2, N5	7
TS2 =>M0, M2, N0	3
TS3 =>M0, MI, M3, M4, N0, N1, N3	7
TS4 => M0, M4, N0, N2	4
TS5 =>M0, Ml, M2, MS, N0, N1, N3, N4	8

Option 6: Send video file N at shift 5 TS	Total Data Blocks
TS0 =>M0, N0, N1, N2, N5	5
TS1 => M0, M1, M3, N0	4
TS2 => M0, M2, N0, N1, N3	5
TS3 =>M0, M1, M3, M4, N0, N2	6
TS4 =>M0, M4, N0, N1, N3, N4	6
TS5 =>M0, M1, M2, M5, N0, N4	6

Applying policy (1), options 2, 4, and 6 have the smallest maximum number of data blocks (i.e., 6 data blocks) sent during any time slot. Applying policy (2), the optimal delivery matrix in this exemplary embodiment is option 4 because option 4 has the smallest sum of data blocks of any reference time slot plus data blocks of neighboring time slots (i.e., 16 data blocks). Thus, optimally for this embodiment, the sending sequence of the data file N should be shifted by three time slots. In an exemplary

embodiment, a three-dimensional delivery matrix is generated for each channel server 104.

When data blocks for each data file are sent in accordance with a delivery matrix, a large number of subscribing clients can access the data file at a random time and the appropriate data blocks of the data file will be timely available to each subscribing client. In the example provided above, assume the duration of a time slot is equal to *5* seconds, the DOD system 100 sends data blocks for data files M and N in accordance with the optimal delivery matrix (i.e., shift delivery sequence of data file N by three time slots) in the following manner:

Time 00:00:00=> M0 N0 N1 N3 N4
Time 00:00:05=> M0 Ml M3 N0 N4
Time 00:00:10=> M0 M2 N0 N1 N2 N5
Time 00:00:15=> M0 Ml M3 M4 N0
Time 00:00:20=> M0 M4 N0 N1 N3
Time 00:00:25=> M0 Ml M2 M5 N0 N2
Time 00:00:30=> M0 N0 N1 N3 N4
Time 00:00:35=> M0 Ml M3 N0 N4
Time 00:00:40=> M0 M2 N0 N1 N2 N5
Time 00:00:45=> M0 M1 M3 M4 N0
Time 00:00:50=> M0 M4 N0 N1 N3
Time 00:00:55=> M0 M1 M2 M5 N0 N2

If at time 00:00:00 a client A selects movie M, the STB 300 at client A receives, stores, plays, and rejects data blocks as follows:

Time 00:00:00=> Receive M0 => play M0, store M0.

Time 00:00:05 => Receive Ml, M3 => play Ml, store M0, M1, M3.

Time 00:00:10 => Receive M2 => play M2, store M0, Ml, M2, M3.

Time 00:00:15 => Receive M4 => play M3, store M0, Ml, M2, M3, M4.

Time 00:00:20 => Receive none => play M4, store M0, M1, M2, M3, M4.

Time 00:00:25 => Recv M5 => play M5, store M0, Ml, M2, M3, M4, M5.

If at time 00:00:10, a client B selects movie M, the STB 300 at client B receives, stores, plays, and rejects data blocks as follows:

Time 00:00:10 => Rcv M0, M2 => play M0, store M0, M2.

Time 00:00:15 => Rcv Ml, M3, M4 => play Ml, store M0, M1, M2, M3,M4.

Time 00:00:20 => Rcv none => play M2, store M0, Ml, M2, M3, M4.

Time 00:00:25 => Rcv M5 => play M3, store M0, M1, M2, M3, M4, M5.

Time 00:00:30 => Rcv none => play M4, store M0, Ml, M2, M3, M4, M5.

Time 00:00:35 => Rcv none => play M5, store M0, Ml, M2, M3, M4, M5.

If at time 00:00:15, a client C selects movie N, the STB 300 of the client C receives, stores, plays, and rejects data blocks as follows:

Time 00:00:15 => Rcv N0 => play N0, store N0.

Time 00:00:20 => Rcv Nl N3 => play N1, store N0, N1, N3.

Time 00:00:25 => Rcv N2 => play N2, store N0, N1, N2, N3.

Time 00:00:30 => Rcv N4 => play N3, store N0, N1, N2, N3, N4.

Time 00:00:35 => Rcv none => play N4, store N0, N1, N2, N3, N4.

Time 00:00:40 => Rcv N5 => play N5, store N0, N1, N2, N3, N4, N5.

If at time 00:00:30, a client D also selects movie N, the STB 300 at the client D receives, stores, plays, and rejects data blocks as follows:

Time 00:00:30=> Rcv N0, N1, N3, N4 => play N0, store N0, N1, N3, N4.

Time 00:00:35=> Rcv none => play N1, store N0, N1, N3, N4.

Time 00:00:40=> Rcv N2, N5 => play N2, store N0, N1, N2, N3, N4, N5.

Time 00:00:45=> Rcv none => play N3, store N0, N1, N2, N3, N4, N5.

Time 00:00:50=> Rcv none => play N4, store N0, N1, N2, N3, N4, N5.

Time 00:00:55=> Rcv none => play N5, store N0, N1, N2, N3, N4, N5.

As shown in the above examples, any combination of clients can at a random time independently select and begin playing any data file provided by the service provider. The above denotation of "Receive" is slightly misleading as the system is always receiving a continuous stream of data blocks determined by the time slot, but at any given point, the receiving STB may only require certain data blocks, having already received and stored the other received data blocks. This need is referred to as "receive" above, but may be more accurately referred to as "non rejected." Therefore, "receive M4" could be termed "reject all but M4" and "receive none" could better be termed "reject all."

What becomes apparent from the examples given above is that available bandwidth is not being fully used during certain time slots. In particular, during at least some time slots there is "idle time" wherein no transmission occurs. This idle time is an inherently ineffective use of available bandwidth. Let's take as an example option 4 shown above wherein two data blocks are transmitted during the respective time slot. In other words, in a time slot having bandwidth suitable for transmitting six data blocks, four data block transmission periods are left idle. Although this is not dramatic in option 4, it becomes more extreme as data files become

thousands of data blocks big. Even using optimal combination protocols for combining data, there may still be significant sections of empty block space.

This empty block space equates to bandwidth which is not being used, and therefore is wasted bandwidth. A goal of this invention is to decrease as much idle time as possible, and therefore one embodiment of the current invention is to perform another step after the scheduling matrix is determined, referred to herein as a decreased idle time scheduling matrix.

An exemplary model of a decreased idle time scheduling matrixes can be explained with reference to the six block scheduling matrix described above, but repeated here for convenience. Idle time during which bandwidth could be utilized to transmit a data block is denoted "<-->" for clarity:

TS0 => blk0, <-->, <-->, <-->

TS1 => blk0, blk1, blk3, <-->

TS2 => blk0, blk2, <-->, <-->

TS3 => blk0, blk1, blk3, blk4

TS4 => blk0, blk4, <-->, <-->

TS5 => blk0, blk1, blk2, blk5

This is shown in graphical form in Fig 5. The scheduling matrix clearly has unused bandwidth in the form of idle time during most time slots. The present invention teaches reduction of this idle time by utilizing constant bandwidth from time slot to time slot. The key to accomplishing decreased idle transmission time through constant bandwidth utilization is an understanding that the delivery sequence of the data blocks must be adhered to, while the exact time slot in which a data block is delivered is not relevant except that the data block must be received prior to or at the time in which it

must be accessed. Accordingly, constant bandwidth utilization is accomplished by transmitting a constant number of data blocks within each time slot according to the delivery sequence set forth by the scheduling matrix and with disregard to the time slot assigned by the scheduling matrix.

In the six block scheduling matrix described above in detail, there is a significant amount of idle time in TS0, TS1, TS2 and TS4. For the sake of example, assume the desired constant bandwidth corresponds to transmission of four data blocks per time slot. Accordingly, the idle time is decreased by moving forward data blocks until four data blocks are scheduled for transmission during each time slot. The procedure for this is to take the next data block in sequence, and move it to the empty space. So for this example, the first block in TS1, blk0, is moved to TS0. The next block in TS1, blk1, is also moved up. Then, since TS0 still has an empty data block space, blk3 from TS1 is also moved up. TS0 then has all of its spaces filled, and now looks like:

TS0 => blk0, blk0, blk1, blk3

Now TS1 and most of TS2 are empty, so the data blocks from TS3 get moved up. Once this sequence is finished, the matrix looks like:

TS0 => blk0, blk0, blk1, blk3
TS1 => blk0, blk2, blk0, blk1
TS2 => blk3, blk4, blk0, blk4
TS3 => blk0, blk1, blk2, blk5
TS4 =>
TS5 =>

This is also shown graphically in Fig. 6. Empty and incomplete time slots, such as TS4 and TS5 in this example, get filled up simply by repeating the original sequence, while still filling up the idle time. Hence the first six time slots would appear as:

TS0 => blk0, blk0, blk1, blk3
TS1 => blk0, blk2, blk0, blk1
TS2 => blk3, blk4, blk0, blk4
TS3 => blk0, blk1, blk2, blk5
TS4 => blk0, blk0, blk1, blk3
TS5 => blk0, blk2, blk0, blk1

The next two time slots in this sequence, TS6 and TS7, would have the same data blocks as TS2 and TS3. Therefore what is actually produced by this process in a new, shorter scheduling matrix, now only four time slots long. Fig. 7 graphically depicts this new repeating matrix created by filling up idle time.

It is obvious from the example given above that as long as the original order is followed, a user can now receive the data file ahead of time in contrast with the on time delivery of the original scheduling matrix. A user may even enter the system mid-time slot and begin using the data as soon as a starting block, blk0, is received.

In this fashion, time slots become largely a computational fiction, as the data blocks become a continuous stream, and at any point in the stream a user may jump onto the system and begin receiving data. As can be seen in

Fig. 8, this additional step is a relatively simple step **510**, performed at what was the end of the procedure **410**.

For simplicity's sake, the above description of FIGS. 4 – 10 dealt with an instance where the selected bandwidth was set to a constant equal to an integer number of data blocks. However, the constant bandwidth need not be equal to an integer number of data blocks. Instead, what is simply required is that the delivery sequence adhere to the sequence developed as in FIG. 8. The data stream generated by the delivery sequence developed in FIG. 8 is then provided to a lower level hardware device (e.g., a network card or the channel server) which controls broadcast of the digital data. Rather than broadcasting an integer number of data blocks, the lower level hardware device will transmit as much data as possible within the bandwidth allocated to the file.

As will be appreciated by those of skill in the art, at the abstraction level of the delivery sequence, one need not worry about the actual transmission of data. Instead, the delivery matrix provides the sequence and the lower level hardware device controls broadcast of data utilizing the allocated bandwidth. Hence an allocated bandwidth which includes a fraction of a data block size can be fully utilized. Once the allocated bandwidth has been utilized, the lower level device will pause broadcast of this particular data file until bandwidth is again available.

GENERAL OPERATION

A service provider can schedule to send a number of data files (e.g., video files) to channel servers 104 prior to broadcasting. The central controlling server 102 calculates and sends to the channel servers 104 three-dimensional delivery matrices (ID, time slot, and data block send order).

During broadcasting, channel servers 104 consult the three-dimensional delivery matrices to send appropriate data blocks in an appropriate order. Each data file is divided into data blocks so that a large number of subscribing clients can separately begin viewing a data file continuously and sequentially at a random time. The size of a data block of a data file is dependent on the duration of a selected time slot and the bit rate of the data stream of the data file. For example, in a constant bit rate MPEG data stream, each data block has a fixed size of: Block Size (MBytes) = BitRate (Mb/s) x TS (see) / 8 (1).

In an exemplary embodiment, a data block size is adjusted to a next higher multiple of a memory cluster size in the local memory 208 of a channel server 104. For example, if a calculated data block length is 720Kbytes according to equation (1) above, then the resulting data block length should be 768Kbytes if the cluster size of the local memory 208 is 64Kbytes. In this embodiment, data blocks should be further divided into multiples of sub-blocks each having the same size as the cluster size. In this example, the data block has twelve sub-blocks of 64KBytes.

A sub-block can be further broken down into data packets. Each data packet contains a packet header and packet data. The packet data length depends on the maximum transfer unit (MTU) of a physical layer where each channel server's CPU sends data. In the preferred embodiment, the total size of the packet header and packet data should be less than the MTU. However, for maximum efficiency, the packet data length should be as long as possible.

In an exemplary embodiment, data in a packet header contains information that permits the subscriber client's STB 300 to decode any received data and determine if the data packet belongs to a selected data file

(e.g., protocol signature, version, ID, or packet type information). The packet header may also contain other information, such as block/sub-block/packet number, packet length, cyclic redundancy check (CRC) and offset in a sub-block, and/or encoding information.

Once received by a channel server 104, data packets are sent to the QAM modulator 206 where another header is added to the data packet to generate a QAM modulated IF output signal. The maximum bit rate output for the QAM modulator 206 is dependent on available bandwidth. For example, for a QAM modulator 206 with 6MHz bandwidth, the maximum bit rate is 5.05 (bit/symbol) x 6 (MHz) = 30.3 Mbit/sec.

The QAM-modulated IF signals are sent to the up-converters 106 to be converted to RF signals suitable for a specific channel (e.g., for CATV channel 80, 559.250MHz and 6MHz bandwidth). For example, if a cable network has high bandwidth (or bit rate), each channel can be used to provide more than one data stream, with each data stream occupying a virtual sub-channel. For example, three MPEG1 data streams can fit into a 6MHz channel using QAM modulation. The output of the up-converters 106 is applied to the combiner/amplifier 108, which sends the combined signal to the transmission medium 110.

In an exemplary embodiment, the total system bandwidth (BW) for transmitting "N" data streams is BW = N x bw, where bw is the required bandwidth per data stream. For example, three MPEG-1 data streams can be transmitted at the same time by a DOCSIS cable channel having a system bandwidth of 30.3 Mbits/sec. because each MPEG-1 data stream occupies 9 Mbits/sec of the system bandwidth.

Typically, bandwidth is consumed regardless of the number of subscribing clients actually accessing the DOD service. Thus, even if no

subscribing client is using the DOD service, bandwidth is still consumed to ensure the on-demand capability of the system.

The STB 300, once turned on, continuously receives and updates a program guide stored in the local memory 308 of a STB 300. In an exemplary embodiment, the STB 300 displays data file information including the latest program guide on a TV screen. Data file information, such as video file information, may include movieID, movie title, description (in multiple languages), category (e.g., action, children), rating (e.g., R, PG13), cable company policy (e.g., price, length of free preview), subscription period, movie poster, and movie preview. In an exemplary embodiment, data file information is sent via a reserved physical channel, such as a channel reserved for firmware update, commercials, and/or emergency information. In another exemplary embodiment, information is sent in a physical channel shared by other data streams.

A subscribing client can view a list of available data files arranged by categories displayed on a television screen. When the client selects one of the available data files, the STB 300 controls its hardware to tune into a corresponding physical channel and/or a virtual sub-channel to start receiving data packets for that data file. The STB 300 examines every data packet header, decodes data in the data packets, and determines if a received data packet should be retained. If the STB 300 determines that a data packet should not be retained, the data packet is discarded. Otherwise, the packet data is saved in the local memory 308 for later retrieval or is temporarily stored, in the buffer memory 310 until it is sent to the decoder 312.

To improve performance efficiency by avoiding frequent read/write into the local memory 308, in an exemplary embodiment, the STB 300 uses a "sliding window" anticipation technique to lock anticipated data blocks in

the memory buffer 310 whenever possible. Data blocks are transferred to the decoder 312 directly out of the memory buffer 310 if a hit in an anticipation window occurs. If an anticipation miss occurs, data blocks are read from the local memory 308 into the memory buffer 310 before the data blocks are transferred to the decoder 312 from the memory buffer 310.

In an exemplary embodiment, the STB 300 responds to subscribing client's commands via infrared (IR) remote control unit buttons, an IR keyboard, or front panel pushbuttons, including buttons to pause, play in slow motion, rewind, zoom and single step. In an exemplary embodiment, if a subscribing client does not input any action for a predetermined period of time (e.g., scrolling program menu, or selecting a category or movie), a scheduled commercial is played automatically. The scheduled commercial is automatically stopped when the subscribing client provides an action (e.g., press a button in a remote control unit). In another exemplary embodiment, the STB 300 can automatically insert commercials while a video is being played. The service provider (e.g., a cable company) can set up a pricing policy that dictates how frequently commercials should interrupt the video being played.

If an emergency information bit is found in a data packet header, the STB 300 pauses any data receiving operation and controls its hardware to tune into the channel reserved for receiving data file information to obtain and decode any emergency information to be displayed on an output screen. In an exemplary embodiment, when the STB 300 is idled, it is tuned to the channel reserved for receiving data file information and is always ready to receive and display any emergency information without delay.

The foregoing examples illustrate certain exemplary embodiments of the invention from which other embodiments, variations, and modifications

will be apparent to those skilled in the art. The invention should therefore not be limited to the particular embodiments discussed above, but rather is defined by the following claims.